Auto-tuning shared memory parallel Sparse BLAS operations in `librsb`-1.2

Michele MARTONE (michele.martone@ipp.mpg.de)

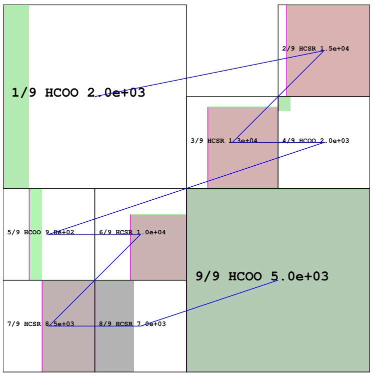
Max Planck Institute for Plasma Physics, Garching bei München, Germany

Intro

- Sparse BLAS (Basic Linear Algebra Subroutines) [2] specifies main kernels for iterative methods:
  - *sparse **M**ultiply by **M**atrix*: "**MM**"
  - *sparse triangular **S**olve by **M**atrix*: "**SM**"
- Focus on **MM**: $C \leftarrow C + \alpha op(A) \times B$, with
  - A has dimensions $m \times k$ and is *sparse* (nnz nonzeroes)
  - $op(A)$ can be either of $\{A, A^T, A^H\}$ (parameter $transA$)
  - *left hand side* (**LHS**) B is $k \times n$, *right hand side* (**RHS**) C is $n \times m$ (n=NRHS), both dense (eventually with *strided* access $incB, incC, ...$)
  - α is scalar
  - either single or double precision, either `real` or `complex`
- `librsb` implements the Sparse BLAS using the **RSB** (**Recursive Sparse Blocks**) data structure [3].
- Hand tuning for each operation variant is impossible.
- We propose **empirical auto-tuning** for `librsb`-1.2.

RSB: Recursive Sparse Blocks

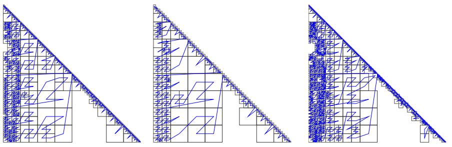
- *Sparse blocks* in COO or CSR [1].
- ...eventually with 16-bit indices ("HCOO" or "HCSR").
- *cache blocks* suitable for *thread parallelism.*
- *Recursive partitioning* of submatrices results in Z-ordered blocks.



- Instance of matrix *bayer02* (ca. $14k \times 14k, 64k$ nonzeroes).
- The black-bordered boxes are *sparse blocks*.
- Greener have fewer nnz than average, redder have more.
- Blocks rows (columns) of **LHS** (**RHS**) range during **MM**.
- Larger submatrices like "**`9/9`**" can have fewer nonzeroes than smaller ones like "**`4/9`**".

Merge / split based autotuning

- Optimal default blocking ?
  - Irregular matrix patterns !
  - Operands (especially $transA$, NRHS) change memory footprint !
- **Empirical auto-tuning**:
  - Given a Sparse BLAS operation, probe for a *better performing* blocking.
  - Search among *slightly coarser* or *finer* ones.



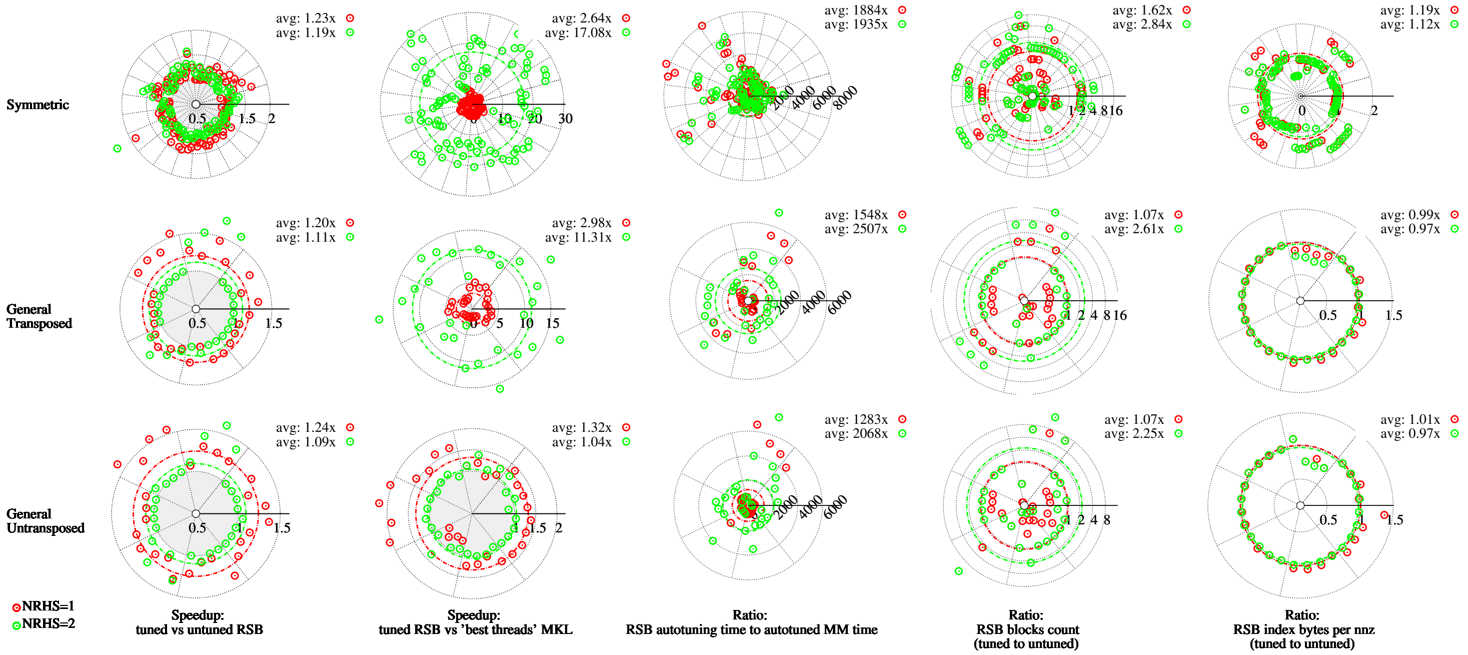
Untuned. Tuned for NRHS=1. Tuned for NRHS=3.

- Tuning example on symmetric matrix *audikw_1*.
- Here only lower triangle, ca. $1M \times 1M, 39M$ nonzeroes.
- On a machine with 256 KB sized L2 cache.
- Left one (625 blocks, avg 491 KB) is before tuning.
- Middle one (271 blocks, avg 1133 KB) after tuning (1.057x speedup, 6436.6 ops to amortize) for **MV** (**MM** with NRHS=1).
- Right one (1319 blocks, avg 233 KB) after tuning (1.050x speedup, 3996.2 ops to amortize) for **MM** with NRHS=3.
- Finer subdivision at NRHS=3 consequence of increased cache occupation of per-block **LHS**/**RHS** operands.

Sparse BLAS autotuning extension

```
! Matrix-Vector Multiply: y ← alpha*op(A)*x+y
call USMV(transA,alpha,A,x,incx,y,incy,istat)
! Tuning request for the next operation
call USSP(A,blas_autotune_next_operation,istat)
! Matrix structure and threads tuning
call USMV(transA,alpha,A,x,incx,y,incy,istat)
...
do ! A is now tuned for y ← alpha*op(A)*x+y
  call USMV(transA,alpha,A,x,incx,y,incy,istat)
...
! Request autotuning again
call USSP(A,blas_autotune_next_operation,istat)
! Now tune for C ← C + alpha * op(A) * B
call USMM(order,transA,nrhs,alpha,A,B,ldB,C,ldC,istat)
! The RSB representation of A is probably different than before USMM
```

Experiment in MM tuning and comparison to MKL



Speedup: tuned vs untuned RSB

Speedup: tuned RSB vs 'best threads' MKL

Ratio: RSB autotuning time to autotuned MM time

Ratio: RSB blocks count (tuned to untuned)

Ratio: RSB index bytes per nnz (tuned to untuned)

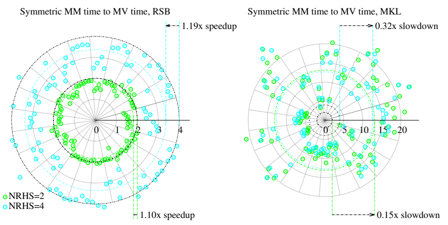
Setup

- `librsb` (`icc -O3 -xAVX`, v15) vs Intel MKL (v.11.2) CSR.
- 2× *"Intel Xeon E5-2680"*, 16 OpenMP threads.
- **MM** with NRHS={**1**,**2**}, four BLAS numerical types.
- 27 matrices in total (as in [3]), including symmetric.

Results Summary

- **Few dozen percent** improvement over untuned, costing **few thousand** operations.
- **Significantly faster than Intel MKL** on symmetric and transposed operation with NRHS=2 (> 1).
- **Autotuning** more effective on **symmetric** and **unsymmetric untransposed** with **NRHS=1**.
- Tuning mostly **subdivided further** for NRHS=2.

Highlight: symmetric MM vs MV performance



RSB Symmetric **MM** performance increases when NRHS>1, while Intel MKL CSR' falls. See here for NRHS=2 and NRHS=4.

Outlook

One may improve via:

- **Reversible in-place merge and split**: no need for copy while tuning.
- Best merge/split choice not obvious: **different merge and split rules**.
- **Non-time efficiency criteria** (e.g. use an energy measuring API when picking **better performing**).

References

[1] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[2] BLAS Forum. Sparse BLAS specification (BLAS specification, chapter 3). Technical report.

[3] M. Martone. Efficient multithreaded untransposed, transposed or symmetric sparse matrix-vector multiplication with the recursive sparse blocks format. *Parallel Computing*, (40):251–270, 2014.

EXASCALE15, Workshop *Sparse Solvers for Exascale: From Building Blocks to Applications*, 23-25 March 2015, Greifswald, Germany